SYSTEM FOR PROGRAMMING A PROGRAMMABLE LOGIC CONTROLLER USING A WEB BROWSER

TECHNICAL FIELD

5   Applicants' invention relates generally to the field of programmable controllers and more particularly to a system for coupling a network of programmable controllers through an internetwork to a monitoring and control device.

RELATED APPLICATIONS

This application is a continuation-in-part of U.S. Patent Application No. 09/303,458, which is a continuation-in-part of U.S. Patent Application No. 08/927,005. This application is also related to the following, commonly assigned application, entitled "Apparatus for Controlling Internetwork Communications," U.S. Patent Application No. 08/926,837. The contents of these Applications are expressly incorporated herein by 15   reference.

BACKGROUND OF THE INVENTION

Remote monitoring and control of systems and processes have taken many forms. In the past, dedicated lines became the most common forms of communication 20   between a control system and a remote location. This has limited application since the control system was not accessible from multiple locations. Modems have made it possible to access the control system from different locations, but these types of systems are generally restricted to downloading and uploading data files. Providing any type of control function between locations is rather limited in this type of environment. 25   Further, an end user generally required a customized interface to access the control system.

With the growth of Internet, and its World Wide Web providing a deliver platform for organizing Internet data through hypertext links, a client server system can be designed that will give each end user the same type of a user friendly interface with the same universal access to services on the Web. The Web is a network of documents 30   called sites or pages stored on server computers throughout the world. Each page will usually contain text, some type of multimedia offerings such as graphic images, video, or audio, and possible hypertext links to other documents. A browser allows a user to read the pages and interact with the choices associated with it. The browser is a 35   graphical software program that sends commands to the Internet Web site and displays

whatever information is available on the page. Various browser programs are commercially available from different manufacturers.

The Internet network employs methods designed to handle thousands of general-purpose computers sharing a single cable, and therefore has no ability to

5 differentiate traffic in terms of its purpose or the criticality of its data. The Internet is no longer a network of computers sharing a single cable, but rather a web of interconnected point to point links involving both general purpose stations and specialized infrastructure components such as routers and firewalls.

The type of personal computer or workstation used by the end user to connect

10 to the Web is of no regard. Communication over the Internet and other networks requires one of several types of protocols. Protocols such as Internet Protocol (IP) provide for file transfers, electronic mail, and other services. A Sun Microsystem programming language known as Java, along with Hypertext Markup Language (HTML) used in designing layouts and graphics for a Web site or page has extended Internet

15 technology such that a web site can be used for dynamic applications, commonly called applets, that can be downloaded and run by the end user. These applets are interpreted and run within a Web browser and have been generally restricted to word processing and similar uses. Downloading and running applets can be slow in comparison to other types of compiled languages. Security rules imposed on a browser and enforced by the

20 underlying JAVA language prevent applets from obtaining certain data from any other device other than the Web server itself.

Programmable logic controllers (PLCs) are widely used in industry and process control. Many manufacturers provide factory automation information using Microsoft Windows and other types of communication networking environments. These networks

25 are usually slow, are not universally accessible and are limited to monitoring and data exchange. Control may be implemented, but since the communication networks are non-deterministic, control is not real time. Specialized industrial networks using proprietary fieldbus alternatives can be very expensive. Conversion products are required to allow information carried over those networks to visible on a general-

30 purpose network. There are significant installation and other deployment costs associated with the existence of such intermediate devices. Firewalls between the Web server and the application are designed to solve problems of security and are not 5 designed for high performance.

Programming the PLCs in these environments has many shortcomings. With

35 many different PLCs possible in an integrated control system, making changes or updates may be difficult and expensive. The programming package is usually sold

separate from the PLC. The user may need to purchase and maintain separate programming packages and may need training for each programming package. The package may have a different look and feel from others within the control system.

It would be desirable to develop an automation control system whereby a user

5   could use general, commercial networks such as the Internet in place of specialized industrial networks to remotely monitor and program automation control devices such as PLCs.

SUMMARY OF THE INVENTION

10   Accordingly, the principal object of the present invention is to provide an interface between an industrial control system and a Web browser coupled to a network such as Internet.

Another object of the present invention is to provide remote access through a web browser to information and data contained in an industrial control system having a

15   Programmable Logic Controller.

A further object of the present invention is to provide remote programmability of a Programmable Logic contained in an industrial control system.

In the preferred embodiment of the invention, the invention allows for easy access over a commercial network such as Internet to information within a

20   programmable logic controller (PLC). Access can be made locally or worldwide using a commercial Web browser. The invention is comprised of a control system of essential elements including, but not limited to a Web interface, a local network, and a network interface to at least one PLC control system running an application program for controlling output devices in response to status of input devices. The Web interface

25   runs Web pages from the Ethernet board coupled directly to the PLC back plane and includes an hypertext transfer protocol (HTTP) interpreter, a PLC back plane driver, a Transmission Control Protocol/Internet Protocol (TCP/IP) stack, and an Ethernet board kernel. The Web interface provides access to the PLC back plane by a user at a remote location through the Internet. The interface translates the industry standard Ethernet,

30   TCP/IP and HTTP protocols used on the Internet into data recognizable to the PLC. Using this interface, the user can retrieve all pertinent data regarding the operation of the PLC, including PLC configuration, Input/Output (I/O) and register status, operating statistics, diagnostics, and distributed I/O configurations. Updates to operating software can also be downloaded through the Internet access. Residing in the PLC is a

35   programming package accessible to a user through this interface which will allow the

user to edit programs controlling the operation of the programmable logic controller system.

Other features and advantages of the invention, which are believed to be novel and nonobvious, will be apparent from the following specification taken in conjunction with the accompanying drawings in which there is shown a preferred embodiment of the invention. Reference is made to the claims for interpreting the full scope of the invention which is not necessarily represented by such embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows an overview block diagram of a typical system illustrating the relationship between an user at a remote location and an Internet Web site used for monitoring a process control system according to the present invention.

Figure 2 is a basic block diagram of the present invention illustrating an Internet interface to a programmable logic controller system.

Figure 3 is a block diagram of the Web server module illustrated in Figure 2 according to the present invention.

Figure 4 is a typical mimic page available to a user at a remote location utilizing a browser which illustrates the present invention for monitoring a programmable controller system.

DETAILED DESCRIPTION

Although this invention is susceptible to embodiments of many different forms, a preferred embodiment will be described and illustrated in detail herein. The present disclosure exemplifies the principles of the invention and is not to be considered a limit to the broader aspects of the invention to the particular embodiment as described.

Figure 1 shows an overview block diagram of typical system illustrating the relationship between a user 2 at a remote location and an Internet web site 4 used for monitoring a process control system 6. The user 2 will have a personal computer (PC) 8 having a commercially available browser 10, such as Netscape Communication's Navigator or Microsoft's Internet Explorer, installed for viewing the contents at the web site 4 by a monitor 12 through a network, such as the Internet 14. The PC provides a remote human-machine interface (HMI) to the process control system 6. Various interconnection services are readily available to provide the physical and electrical interconnection from the PC to the Internet 14 itself. The Internet 14 is a collection of independent world wide communication networks that are interconnected to each other and function as a single connectionless entity. Communication is based on a

-4-

client-server basis, using a number of established protocols that allow for communication and file transfers between the client and the server. The most widely used protocol is Internet Protocol (IP).

The web site 4 includes a network interface 16 having a unique Internet address 18, a server 20, and an application program 22. The server 20 acts as a hypertext transfer protocol (HTTP) interpreter which uses Transmission Control Protocol (TCP) in conjunction with Internet Protocol, through a Transmission Control Protocol/Internet Protocol (TCP/IP) stack 24 to interact with the network interface 16 and the application program 22. This enables the data transfer between the application program 22 and the user 2 through the Internet 14. The application program provides data from the process control system 6. This data can be used to monitor the control process by the user 2 at the remote location. The TCP/IP stack 24 enables data transfers over the Internet 14 between the user 2 and the web site 4 as required for the various layers specified by the IP protocol.

The user 2 can connect to the Internet 14 using one of a number of Internet service providers and will enter the address of the web site 4 when connected. The web site 4 will display a theme page which may contain text, some type of multimedia offerings such as graphic images, video, or audio, and possible hypertext links to other documents. The browser 10 will allow the user 2 to read the page and interact with the choices associated with it. The browser 10 will send commands to the web site 4 which will use the application program 22 to display whatever information is available from the process control system 6. The browser 10 functions as a remote human-machine interface or HMI control of the process control system as will be detailed below.

Figure 2 shows a basic block diagram of the present invention illustrating the Internet interface to a programmable logic controller system. The web site 4 includes the network interface 16 having a unique Internet address 18 and a web server 30. The web server 30 provides the home page for the website. A firewall or security for the overall system can be included in the web server 30, but is generally maintained as part of the network interface 16. In addition to providing security for various pages at the site, the user can disable the web server 30. A password and user list is provided in initial configuration files stored in the web server 30 that are downloaded from a remote server. Protection of the configuration file is then provided by the remote server and the web server 30 through the password and the user list. The web server 30 provides a direct connection for a programmable logic controller (PLC) 32 to the Internet 14 by plugging the web server 30 into its back plane 34. The web server 30 provides both a client and server interface. All signals between the PLC 32 and the web server 30 are

through the back plane 34 rather than over a set of cables which would normally have to be coupled to input/output modules that are themselves plugged into the back plane 34. The back plane signals include addressing, control, data, and power. The client interface allows a user to send commands to a remote node over the Internet and the

5 server interface allows for processing commands that originated from a remote node. Controlling the PLC 32 from a remote HMI, essentially on a real time basis is possible by controlling the data flow through the web server 30.

Associated with the PLC 32 are its application programs 36, dual port memory 38 and Input/Output (I/O) modules 40. The application programs 36 include a ladder

10 logic program for controlling the I/O modules 40. The web server 30 functions as a node on a TCP/IP network 42 allowing it to send commands to the PLC 32 and receive the response. Although the TCP/IP network 42 in the preferred embodiment is an Ethernet network, other high level protocols could be used. Using a web browser at a remote location through the Internet 14, a user can control and view configuration information

15 of the PLC 32.

A programming package 33, using either Java or HTML resides in the PLC 32. It is based on Ethernet and Web technology and allows editing of the application programs 36 using the personal computer 8 by the user 2. The application programs 36 stored as either a ladder logic or an IEC 1131 language program, are viewed as PLC

20 programs within the PLC 32, accessible using either the standard HTTP or custom PLC protocols that run over TCP/IP. The information needed to edit the PLC 32, including web pages 35 for the editing procedures, reside in the PLC 32 itself. The programming package 33 converts these application programs 36 to and from formats needed for editing with a browser. The programming package 33 also includes symbol editors 37

25 and language editors 39.

The user 2 will be able to view and edit the PLC application programs 36 from anywhere a browser is available. A firewall or security for the overall system, maintained as part of the network interface 16 that prevents unauthorized users from accessing the programming package 33.

30 In addition to providing the ability to edit the application programs 36, by providing a File Transfer Protocol (FTP) based view of the application programs 36, file backup and restore can be handled through standard programming tools, allowing the user to manage the application programs 36 directly and efficiently and to also use standard revision control software to manage the application programs 36. Further,

35 since all the symbols are stored within the PLC, any personal computer can be used to edit the application programs 36.

The web server 30 is shown in greater detail in Figure 3. Various components provide the required connectivity to perform its functionality. A real time operating system 44 controls the interaction between the components. The operating system 44 allocates processor time on a central processor unit (CPU) 46 to various tasks, provides

5   memory management, and provides a set of message services and signal services. The message and signal services allow for communication between tasks, and between drivers and a task.

Connection to the TCP/IP network 42 is through an Ethernet driver 48 which transmits and receives messages at a rate of 10 Mbs (megabits per second) or faster

10   via an Ethernet communication chip 49, such as, in the case of 100 Mbs Ethernet, a Crystal CS8952 available from Cirrus Logic, Inc. It is contemplated that in the future Ethernet communication chips capable of speeds of 1 Gbs (gigabits per second) and faster can be used. The physical connection over which communication is facilitated can be over a fiber optic cable or a twisted pair-type copper wire. In the case of 10 Mbs

15   Ethernet over twisted pair-type copper wire, the copper wire and wire connections must conform to at least the Category 5 copper wire standard.

Alternatively, the TCP/IP stack 24 could be replaced by a dual TCP/IP stack. The dual TCP/IP stack comprises a first TCP/IP stack which provides support for a broad range of TCP/IP messages. The second TCP/IP stack is a "short stack" which

20   manages high priority control messages between the PLC 32 and I/O modules 40. For outgoing TCP/IP messages, the appropriate TCP/IP stack would be chosen by an application program 36 associated with the PLC 32. On incoming TCP/IP messages, the TCP/IP message would be intercepted and examined to see if it is a TCP/IP message that is intended for the Mod bus on Ethernet protocol. If it is a TCP/IP

25   message intended for the Modbus on Ethernet protocol, then the message is delivered to the second stack. If it is not a TCP/IP message intended for the Modbus on Ethernet protocol, the first TCP/IP stack handles the message. In this manner, TCP/IP messages intended for the Modbus on Ethernet protocol are handled more quickly and efficiently than when a single TCP/IP stack is implemented.

30   The web server will have a unique global address 18, allowing it to be addressed by other devices on the network. The Ethernet driver 48 manages transmit 50 and receive 51 buffers in memory 52, and interfaces with the Ethernet communication chip 49. The transmit 50 and receive 51 buffers are shared both by the Ethernet communication chip 49 and the Ethernet driver 48. The Ethernet driver 48 also provides

35   a transmit request interface and a receive indication interface to a TCP/IP stack 54. The Ethernet communication chip 49 provides a transmit queue interface, a receive queue

-7-

interface, and generates interrupts on completion of transmitting a message, and on receiving a new message. The Ethernet driver 46 places receive buffers in the receive queue. In the interrupt routine, the Ethernet driver 46 examines the receive queue. If any messages are in the receive queue, it passes the receive buffer to the TCP/IP stack

5   54. The TCP/IP stack 54 copies the buffer, and sometime later calls the Ethernet driver 48 to return the buffer and place the returned buffer back into the receive queue.

The TCP/IP stack 54 calls the Ethernet driver 48 to transmit a message. The Ethernet driver 46 attempts to allocate a buffer from the shared memory 52. If it succeeds, it copies the message into the buffer, and places the buffer into the Ethernet

10   communication chip 49 transmit queue. If there is no transmit buffer, then the driver drops the transmit message. In the interrupt routine, the Ethernet driver 48 examines the transmit queue, and frees the transmitted buffers.

The TCP/IP network 42 allows special MSTR (master) functions that allow nodes on the network to initiate message transactions. These MSTR functions include

15   reading and writing data and are used for commands and responses. They allow programs running in the PLC 32 to send commands to a remote node on the TCP/IP network 42 and receive the responses. A back plane driver 56 sends commands and receives the response to the PLC 32 over the back plane 34.

The back plane driver 56 receives request from the PLC's ladder logic MSTR

20   blocks stored in its memory 38. When a response is available, the back plane driver 56 passes it back to the MSTR block. The back plane driver 56 provides a client task 58, a FTP task 59, and server task 60 to the applications. The server task 60 interface allows an application to issue a request command to the PLC's 32 executive program and receive its response. The client task 58 allows an application to receive a new MSTR

25   request and pass back the response to the ladder logic program. The FTP task 59 provides a FTP based view of the application programs 36, allowing the user to manage the application programs 36 directly and efficiently and to also use standard revision control software to manage the application programs 36.

The server task 60 uses a queuing mechanism and call back functions. An

30   application queues both the request and the call back function associated with the request. When the back plane driver 56 services the request in its interrupt routine, it calls the associated call back function. The response and the original request is passed to the call back function. The call back function can call an operating routine to either pass a message or signal the application.

35   The client task 58 also uses queues and call back functions. The client application queues both an indication request on queue and a call back function

associated with the request. When the back plane driver 56 detects a new MSTR block request in its interrupt routine, it calls the associated call back function. The request is passed into the call back function. The call back function can call an operating system routine to either pass a message or signal the application. If the back plane driver 56

5 detects that the MSTR block has been aborted, or is no longer being solved, it calls an user supplied associated abort call back function. The application calls a routine to pass the IVISTR response and an associated call back routine to the driver. Sometime later, the driver passes back the response to the ladder logic program in its interrupt service routine, and then calls the user supplied call back function.

10 The PLC 32 interfaces with the web server 30 hardware via the dual port memory 38. It reads and writes to the dual port memory 38 using an ASIC chip. Writing to a specified location will cause an interrupt. The PLC 32 first writes a message in the dual port memory 38, and then causes an interrupt. The message indicates a type of command. One type indicates that a MSTR block is being solved. Other types are used

15 for passing requests to the PLC 32 and obtaining the responses to the requests. After the PLC 32 passes the message, it polls the dual port memory 38 for commands placed by the back plane driver 56. These commands are read memory, write memory, and processing is complete. The back plane driver 56 uses state machines to process the IVISTR interrupts. The maximum number of active MSTR blocks is set at four in the

20 present invention, requiring four state machines. When the back plane driver 56 receives a IVISTR interrupt, it attempts to find an associated state machine that matches with the MSTR block. If there are already four outstanding transactions, no more are available, and the back plane driver 56 will set the MSTR's outputs to false. If a state machine is found, the back plane driver 56 determines if it is a new transaction,

25 an outstanding transaction, or a response is available. If it is a new transaction it copies the request and calls the application's associated call back routine. If it's an outstanding transaction, it indicates to the ladder logic program that the IVISTR block is still busy. If a response is available, the back plane driver 56 copies the response, sets either the MSTR's completion or error output, and calls the application's call back routine.

30 Two interrupts are used for processing a request. On the first interrupt, called the preport interrupt, the back plane driver 56 copies the request into a data structure located in the PLC's 32 dual memory 38. On the second interrupt, called the end of scan interrupt, the back plane driver 56 copies the response from the controller's data 5 structure into the user's buffer. It then calls the user's associated call back function.

35 The request for accessing the PLC's 32 registers is processed by the back plane driver 56 and is not sent to the PLC's executive program for processing. The back plane

driver 56 determines the memory location in the memory 38 of the registers the PLC 32. At an end of scan interrupt, the back plane driver 56 processes the read/write register requests by sending commands via the dual port memory 38 to the PLC 32 to read or write the locations containing the registers. The back plane driver 56 will service a
5   maximum of four read/write register requests at the end of a scan interrupt.

A client task 58 interfaces with the TCP/IP stack 54, the back plane driver 56 and uses the operating system 44 message services. It processes the IVISTR request. When the client task 58 receives a IVISTR request from the back plane driver 56, it passes the request to the TCP/IP stack 54. When the TCP/IP stack 54 returns a
10   response to the client task 58, it passes the response to the back plane driver 56. The TCP/IP stack 54 provides a Berkeley TCP/IP interface and a signal extension. The signal extension calls a user supplied function which passes in a socket number, a task ID, and an event. The signal function calls the operating system 44 to send a message to the task indicated by the task ID. It sends a message either to the client task 58 or
15   server task 60. The client task 58 posts request indications to the back plane driver 56, and the associated call back routine calls the operating system 44 to send a message to the client task 58 for a new MSTR transaction.

The client task 58 manages multiple outstanding MSTR transactions using the state machines. There is a linked list of connection state machines. The connection
20   state machines are used for establishing connection and closing connections. In addition each connection state machine contains a list of transaction state machines. Each transaction machine on the connection state machine represents a transaction to a node represented by the connection machine. The transaction machines are used to send a request, and process the response. The client task 58 enters a loop after
25   performing initialization. It calls the operating system 44 to receive a message. The operating system will blink the client task 58 until there is a message or until there is a time out. It either receives a message from the TCP/IP stack 54, from a IVISTR call back routine, or it times out. It process the message or the time out and then reenters the loop. If the message received from the operating system 44 is a new IVISTR
30   request, the client task will obtain a connection state machine, and places a new transaction machine at end of the list of the connection state machine's list. At this point the transaction machine will attempt to transmit the message. It may not be possible to transmit the message because no connection has been established, or the because the remote side may have applied flow control.
35   If the message received from the operating system 44 is a TCP/IP event, the client task 58 finds the associated connection machine and determines if the TCP/IP

-10-

event is an accepted connection, an aborted connection, or a received data event. Based on the connection state, and the transaction machine's state, the client task 58 processes the message to advance the transactions if there are any. Receiving data for the IVISTR responses may occur over several TCP/IP events, and the transaction state machine assembles the data into a response. When the client task 58 requests the TCP/IP stack to transmit a message, not all of the message may be transmitted. This occurs when the remote node is flow controlled, which is explained below. If the call to the operating system 44 to receive a message returns with a time out, or if there is a message, the client task 58 searches the list of connection machines that are flowed controlled. For each flow controlled connection, it tries to advance the transaction state machines on the connection state machine list that are flow controlled.

The server task 60 processes a request originating from the user at the remote location. The server task 60 interfaces with the back plane driver 56, the TCP/IP stack 54, and the operating system's 44 message services. The server task 60 posts requests to the back plane driver 56, and an associated call back routine uses the operating system 44 message services to send the response to the server task 60. A TCP/IP stack 54 signal function also uses the operating system's 44 send service to send an TCP/IP event to the server task 60. The server task 60 can handle multiple transactions and connections. Like the client task 58, it maintains a list of connection machines, and each connection machine contains a list of transaction machines. The connection machines are for managing the connection and the transaction machines manage the incoming requests and responses.

The server task 60 enters a loop after performing initialization. It calls the operating systems 44 to receive a message. The operating systems 44 block the server task 60 until there is a message or until it times out. It either receives a message from the TCP/IP task's 54 signal handler, from the back plane driver 56 or it times out. It processes the message or the time and reenters the loop. If the message received from the operating systems 44 is from the TCP/IP task's 54 signal handler, the server task 60 determines if the event is a connection request, a close socket event, or a receive data event. Based on the TCP/IP event, the server task 60 uses the connection machine and transaction machine to advance the transaction. Received data for a request may occur over several receive data events, and the transaction machine assembles the events into a request message. When the response message is received from the operating system 44, the server task 60 finds the connection and transaction machine in order to send the response.

When the server task 60 requests the TCP/IP stack 54 to transmit a message, not all of the message may be transmitted. This occurs when the remote node is flow controlled. If the call to the operating system 44 is to receive a message returns with a time out, or if there is a message, the server task 54 searches the list of connection machines that are flowed controlled. For each flow controlled connection, it tries to advance the transaction state machines on the connection state machine list that are flow controlled.

After the server task 60 has parsed the header of an incoming request, it attempts to allocate a structure to pass the request to the back plane driver 56. If the server task is already processing a predetermined number of outstanding requests, the attempt fails, the connection is placed into a blocked state, and the body of the request is not read from the TCP/IP stack 54. As a result the TCP/IP stack may apply flow control to the remote node. When one of the other requests is complete, the free data structure event causes a blocked connection machine to continue processing the incoming Modbus request.

The HTTP task 62 interfaces with the TCP/IP stack 54, and the back plane driver 56. The HTTP server task 62 receives a HTTP request from the TCP/IP stack 54. To process the request, it may access the PLC 32 through the back plane driver 56 and back plane 34. The HTTP server task 62 sends back the response over the TCP/IP stack 54. The framework is supplied by the operating system 44. The framework creates the HTTP task, accepts connection, and parses the HTTP request. After parsing the request, it calls the operating system 44 to process the request. Processing the request involves determining the request type and processing the actual request. The different request types allow a user to acquire a snapshot of the PLC 32 operations by allowing a view of various registers within the PLC 32 and dual memory 38. These request types also include display of the PLC 32 configuration, remote and distributed I/O and module health statistics, display registers, back plane configuration, Ethernet statistics, editing the PLC program, and others, as shown in Table 1:

| Show the home page |
| Show the programmable logic controller's configuration |
| Show the Ethernet statistics |
| Show the read register request page |
| Show the 4x registers |
| Show the racks attached to the controllers back plane |
| Send an image. The different images are gif files that are displayed on the various pages |
| Show the remote I/O statistics |

Show the list of configured remote I/O drops
Show a remote I/O rack's configuration and health
Show a remote I/O drop's communication statistics
Show the I/O reference values of a remote I/O module
Show a list of configured distributed I/O nodes
Show the configuration and the health of a distributed I/O
node
Show the I/O reference values of a distributed I/O module
Edit the PLC Program

TABLE 1

The home page contains hyperlinks to many pages of data. The configuration page will display the configuration of PLC 32. The remote I/O and distributed I/O

5   module health status pages are a series of linked pages. The first page displays the communication health statistics at the Remote I/O and Distributed I/O head and contains a link to a configured drop page. The configured drop page displays a table containing drop numbers which are linked to a drop status page and rack numbers which are linked to the drop and rack configuration pages. Two tables are included in

10   the drop status page, one for showing the communication status of the drop and the other for showing which racks are populated with the I/O modules. The drop and rack configuration page displays the I/O modules, their health, and slot location for the given rack. From a selected module, a user can view its input and output values. Register data is displayed in a template having a form and a table, with the user entering an

15   address and a length. The table will display the register's values A table showing option modules and their slot location is displayed on the back plane configuration page. The data appearing on the pages is static but can be automatically updated at pre-selected times.

The operating system 44 processes these requests and responds by sending

20   HTTP messages through the TCP/IP stack 54. Processing some of these requests involves reading the PLC's traffic cop, registers, coils, or various page zero locations where statistics are kept. To perform these reads, the operating system 44 sends a request to the back plane driver 56 and uses an event signal mechanism and event flags to determine when the request is complete. After sending the request to the back

25   plane driver 56, the operating system 44 waits for an event flag to be sent. When the back plane driver completes the request, the back plane driver 56 calls a call back

routine, which sets the event. The operating system 44 then resumes processing the request.

A mimic page which represents some of the hardware physically connected to a programmable logic controller system can be constructed utilizing the graphical editing technique that is part of the present invention. The present invention allows a user at a remote location, using a browser, to create, view, modify, and delete the mimic page and actually control various components illustrated in the mimic page. Figure 4 shows a simple motor start-stop control in ladder logic diagram form that could be available as a mimic page to the user. Pushing a motor start push button 150 will cause a motor start relay 152 to energize through a normally closed stop push button 154 and a normally closed overload contact 156. Auxiliary motor start contact 158 will latch relay 152 after the start push button 150 is released and pilot light 160 will illuminate. Auxiliary motor start contact 162 will provide power to pump motor 164 which will remain running until stop push button 154 is depressed or overload relay 166 detects an overload condition. In this example, start push button 150, stop push button 154, overload contact 156, auxiliary motor start contacts 158 and 162, and overload relay 166 are inputs to the programmable logic controller system. Relay 152, pilot light 160, and pump motor 164 are outputs. The PLC will have the registers containing the animation data for the inputs and outputs. An application program in the PLC will respond to the inputs to control the outputs.

A user at a remote location will browse the Internet for the home page of the installation of the programmable logic controller system. The PLC will have other control functions as well and if the user has the necessary authorizations, various options will become available. The home page will allow the user to acquire a snapshot of the PLC operations by allowing a view of various pages that will allow access to registers within the PLC. Other pages will also include displays of the PLC's configuration, remote and distributed I/O modules health statistics, display registers, back plane configuration, Ethernet statistics and others as shown previously shown in Table 1.

The mimic diagram page will be called up on a browser screen which will allow the user to view the status of the system. The mimic diagram's light 160, relay 152, contacts 158, 162, and pump motor 164 will be updated to correspond to the state of the actual devices. The rates of the inputs and outputs will then be shown on the ladder diagram which will be automatically updated as they are changed. Through the use of applets representing the start 150 and stop 154 buttons, the user could manually control start and stopping of the motor by using a mouse or keyboard to position a cursor and "clicking" on either the start 168 or stop 170 boxes.

While the specific embodiments have been illustrated and described, numerous modifications are possible without departing from the scope or spirit of the invention.